Exhibit C3

RINGMASTER RELEASE 1.1 FUNCTIONAL SPECIFICATION

PROJECT NAME "NEED IT"

Revision 0.10

[redacted]

AUTHORS: RingMaster Engineering

Revision	Who	Date	Description
0.1	Allan	[redacted]	Original, incorporated various mini-func specs into this one
0.2	Sudhir		Added contents to RF planning section
0.3	Sudhir		Added UI screen shot for WPA support and updated impact of 11g on channel assignment
0.4	Sudhir		Added UI screen shots which will be affected by 11g in an incremental way
0.5	Kishan		Added HP Openview integration section
0.6	Allan		Add mob acls placeholder and cleanup rest
0.7	Sudhir		Changes to 11g and WPA based on internal review
0.8	Yun		Added rules implemented in 1.1 and addtional config support
0.9	Jeff/Sudhir		Added Event Viewer enhancements. Removed WPA2 as a possible security mode
0.10	Sudhir		Changes to 11g based on 11g review meeting

1 INTRODUCTION

1.1 GOALS AND SCOPE

The goal of this document is provide a functional specification of the additional features and functions for RingMaster 1.1.

1.2 OVERVIEW

RingMaster V1.1 is a minor update to the field that includes the following high-level features:

- HP OpenView Integration
- Solaris OS Support
- MSS 1.1 Support
    - WPA
    - 11G Support
    - Boot/Upgrade changes
        - Not yet understood if this impacts RM or not
    - Mobility ACLs
        - Not yet understood the full impact of these changes

In addition to these new features, RingMaster functionality will also be changed in the following areas:

- Transaction Management
    - To improve scalability and performance (i.e. MROW)
- Versioning
    - Fundamental to support 1.0 and 1.1 MSS versions at the same time in RingMaster.
- Additional Rules
    - Including some new rules we missed or deferred in 1.0
- Bug Fixes deferred from 1.0 or found in FCS version of RingMaster

2 802.11G SUPPORT

802.11g is a RF technology that works on the same frequency band as 802.11b. it is similar to 802.11b in channel numbers allowed for the technology. It uses a different modulation to provide the high over-the-air data rate. It is possible for a 802.11g radio accept 802.11b client. This degrades the 802.11g performance.

2.1.1 INFORMATION MODEL

2.1.1.1 RADIO

Radio Type: A radio can be of the types 802.11b, 802.11a, **and 802.11g.**

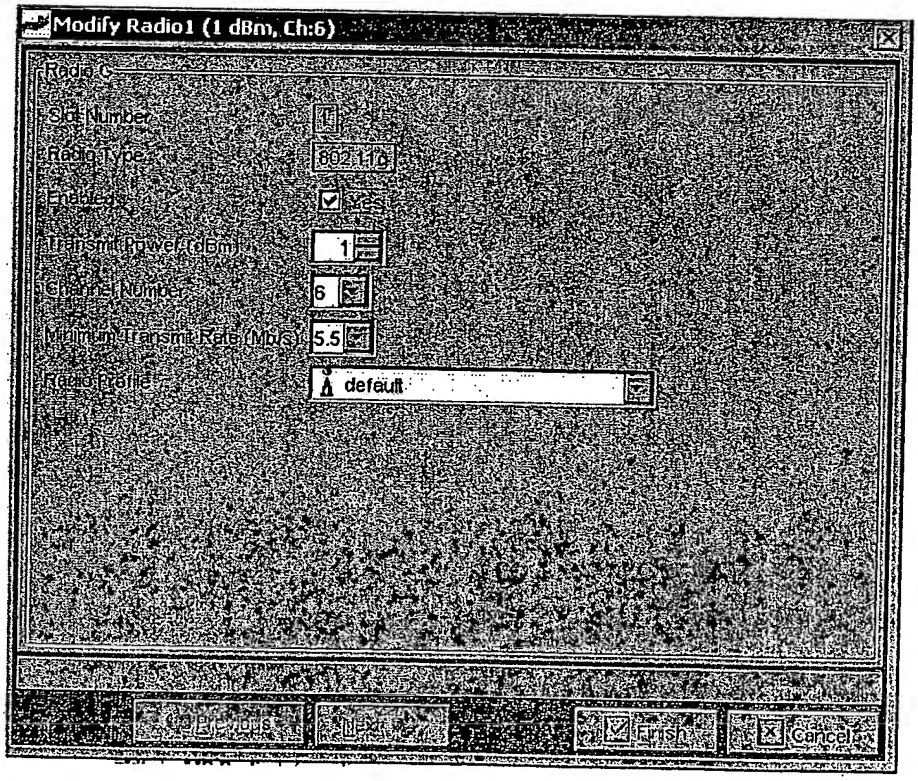
Channel Numbers:

802.11g uses the same channel set as 802.11b. However, it is possible for some countries not to support it.

Transmit Power:

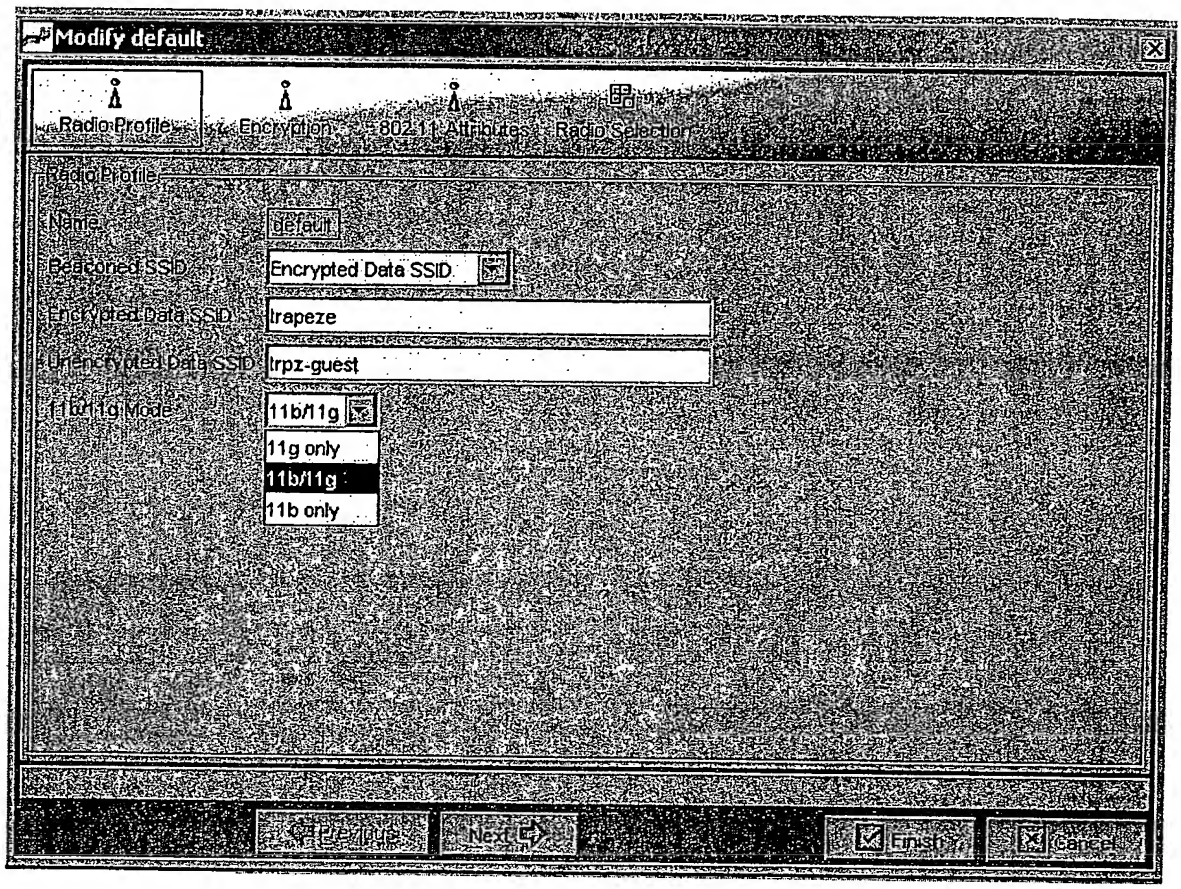
From the regulatory domain point-of-view, 802.11g can use powers similar to 802.11b. The variation of transmit power for a 802.11g radio will depend entirely on the chip-set used. This information will also be product management.

Action Item: (Product Management): To provide Country specific information with related to 802.11g support



2.1.1.2 RADIO PROFILE

Force 11g only: This attribute is required for 11g radio to allow / restrict 11b clients. When checked, it will be in "pureG" mode and when unchecked, it will be in "mixedBG" mode. When in mixedBG mode, the radio can accept 11b clients and also listen to 11b beacons.



2.1.1.3 MP-MODELS

The following new models will be available to support 802.11g. the actual model number cannot be specified as yet.

- Single-radio-802.11(a, b, g)-only (In this document, refered to as MP-241)
- Dual-radio-802.11(a, b/g) (In this document, refered to as MP-252) since the BG radio can be soft configured as 11b or 11g, the radio type attribute will qualify this information.

With the possible introduction of these two modules, RingMaster will allow possible MP models

MP Model	MP type	Radio Type (MP subtype)
MP-122	MP-122	None

MP-101	MP-101	11a or 11b
MP-241	MP-241	11a or 11g
MP-252	MP-252	None

2.1.1.4 RECEIVER SENSITIVITY:

The receiver sensitivity of the radio in 802.11 g will not be the same as 802.11b due to the variation in possible data rates. The sensitivity for 11g radio is shown in the following table. As a comparison, the same for 11b/11a are also shown.

Data Rate (Mb/s)	802.11a	802.11b	802.11g
1		-92	
2		-90	
5.5		-89	
6	-88		-88
9	-86		-86
11		-87	
12	-85		-85
18	-83		-83
24	-80		-80
36	-76		-76
48	-71		-71
54	-70		-70

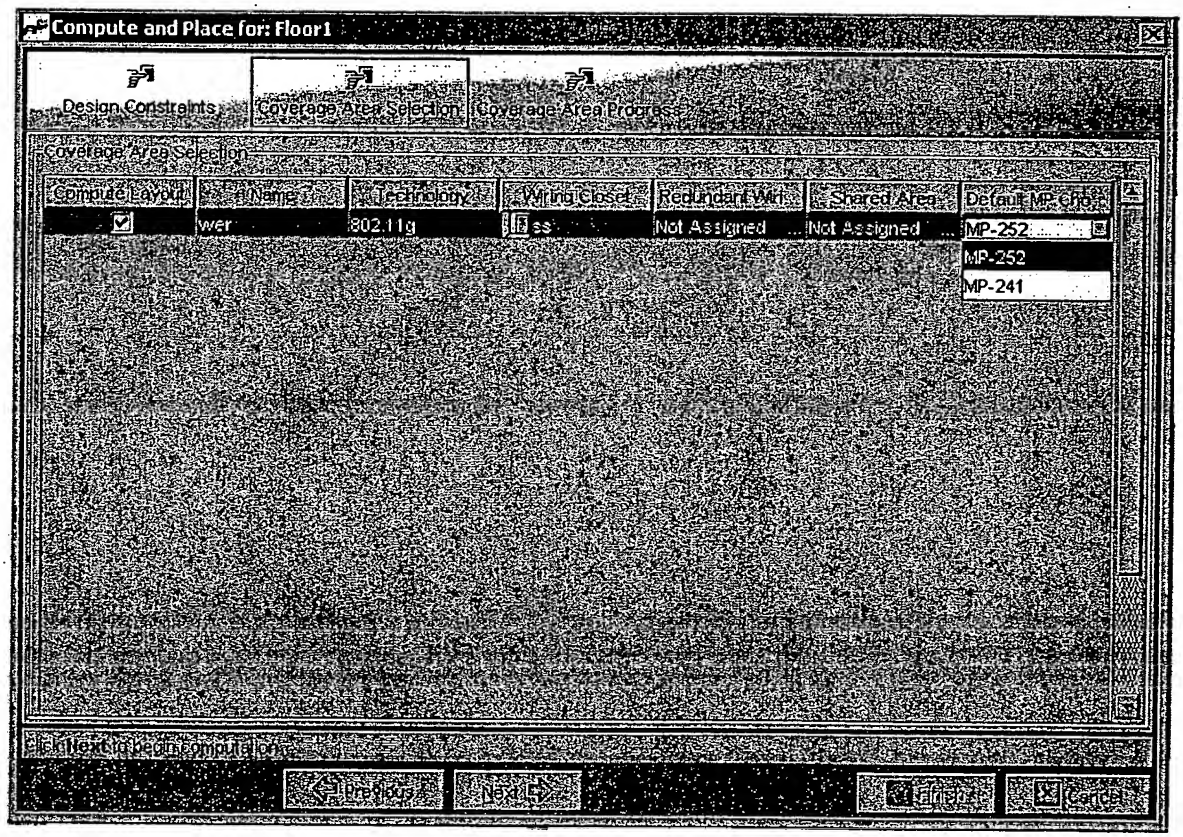
2.1.1.5 DESIGN CONSTRAINTS

There is a design constrant that the user is allowed to select:- Default MP Choice. Currently, it has choices of MP models. With the introduction of 11g, it is no longer the entire floor option as all models do not fit all combinations of technologies. Therefore, this constraint will become an attribute on Coverage Area and will also be allowed to change in 2nd page (coverage Area selection) of Compute and Place wizard.

The choices that will be available for coverage areas are as follows:

Area technology	Choices	Default Choice
11a, unshared	All choices	MP-241
11b, unshared	All choices	MP-241
11g, unshared	New models only	MP-241
11a and 11b, shared	Dual Radio models only	MP-252
11a and 11g, shared	New dual-radio model only	MP-252

The 2nd page in compute and place operation will look as follows:



2.1.1.6 COVERAGE AREA

When creating a coverage area, the user can choose from the following in addition to the existing choices:

- 802.11g only
- 802.11a and 802.11g

Coverage Area will have an additional attribute to allow/disallow 802.11b clients. This information will be rippled to all the associated 11g radios in the coverage area.

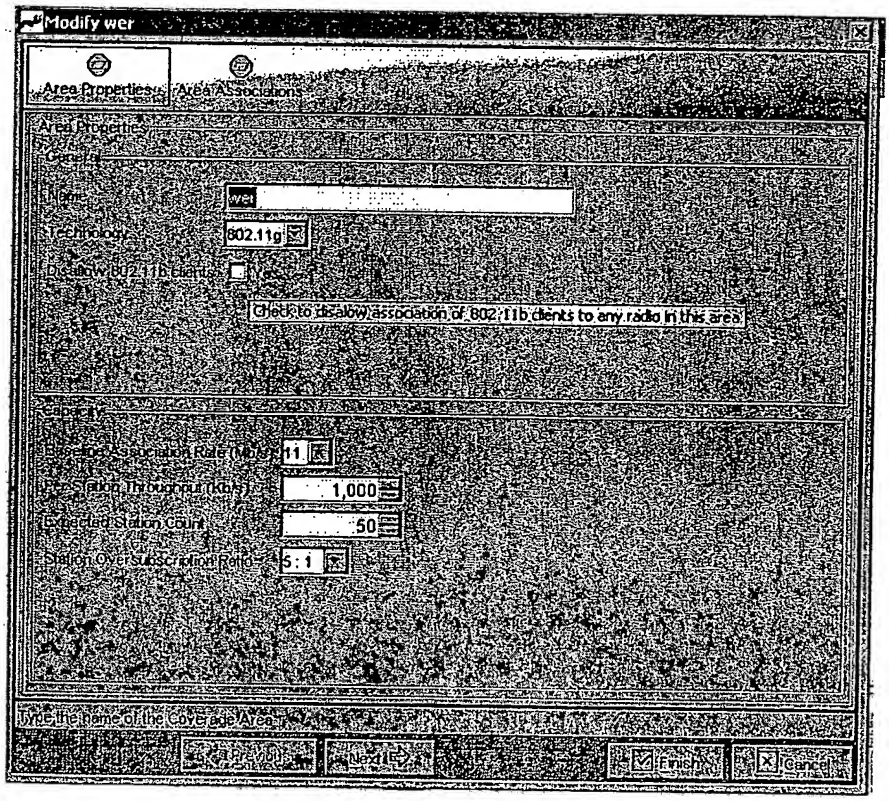
ForceG: This attribute will be provided to allow the user to force 11g mode on the radios associated with a 11g coverage area. This attribute will be enabled and visible only for 11g coverage area.

Radio Profile: the user will be able to choose a radio profile that will be applicable to all the radios of the given coverage area. If the selected radio profile is not found in the configuration of the device of any radio, the radio profile configuration is applied to that device.

i. The list of radio profiles that will be available will depend on the mobility domain associated with the coverage area. It will show all the radio profiles that are policies

ii. The user will be able to create a new radio profile policy from the area wizard.

iii. Any changes to the radio profile property will apply to all radios associated to the coverage area when the modify wizard is finished.



2.1.1.7 RF OBSTACLES

The Attenuation factor of an RF obstacle is same in 11b and 11g as they share the frequency band. The Label of the attenuation factor will reflect the same.

2.1.1.8 CHANNEL SET

Similar to 11b, 11g needs a channel set selection at the network plan level. However, since they share the same frequency band, the selection of the channel set must be same for 11b and 11g. Hence, the label will reflect that this channel set is for both 11b and 11g.

2.1.2 MP COUNT COMPUTATION

As 802.11g radio can accept 802.11b clients, it becomes critical in MP count computation based on capacity that this behavior is known. This behavior also depends on the final chip-set that is selected. Going on the assumption that this is possible, this information should be known before capacity based computation is performed.

Here are the values of some constants used in the computation logic:

Constant or Attribute	11a	11b	11g
Loss Margin	5dB	10dB	5dB
Baseline association rate	36 Mbps	11 Mbps	24 Mbps
Minimum transmit	18 Mbps	5.5 Mbps	12 Mbps
Baseline association rate for 11g in mixedBG mode			*Will not be more 11 Mbps*
Minimum transmit rate for 11g in mixedBG mode			*Will not be more 5.5 Mbps*

Action Item: (Product Management) to provide the defaults of baseline association rate for 11g.

This behavior does not impact the coverage based computation as in the empirical model, the frequency for both 802.11b and 802.11g is the same. All 11g constants will be used for computation. Therefore, the maximum receiver sensitivity will be used based on the association rate specified.

2.1.3 MP PLACEMENT COMPUTATION

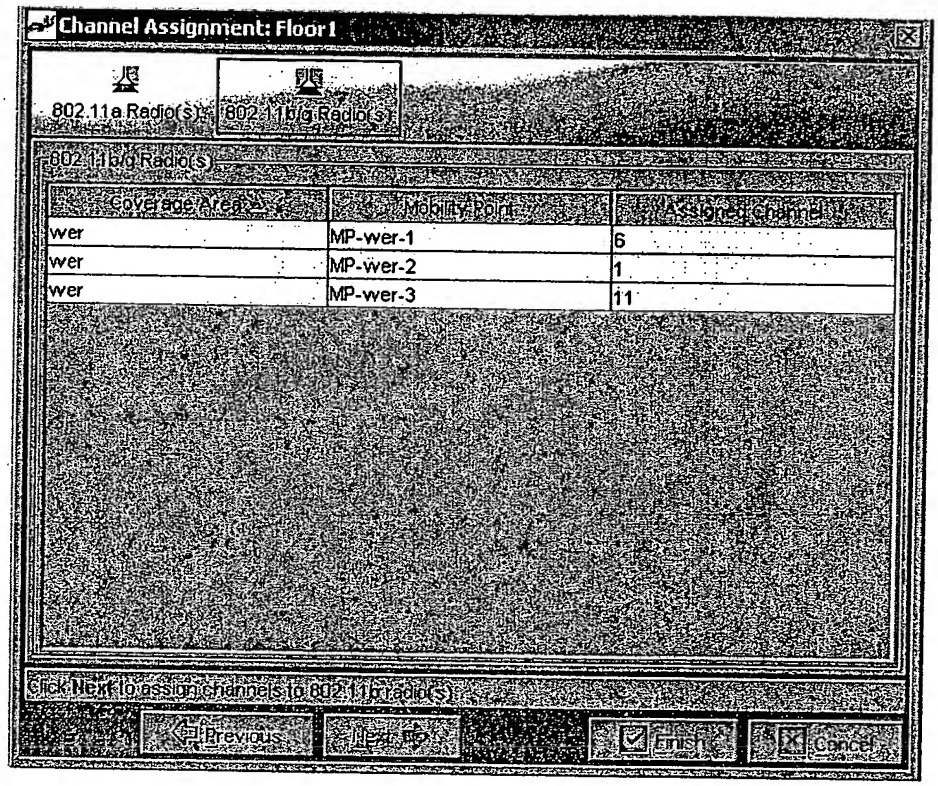
There is no impact to the placement of MPs with introduction of 802.11g

2.1.4 OPTIMAL POWER COMPUTATION

There is no impact to optimal power computation of MPs with introduction of 802.11g.

2.1.5 CHANNEL ASSIGNMENT

11g uses the same channel numbers as 11b. So, when channel assignment is performed for the entire floor, all 11b and 11g radios will be considered together to reduce co-channel interference. The UI will show all the 11g radios in the current 11b page.



2.1.6 RF COVERAGE

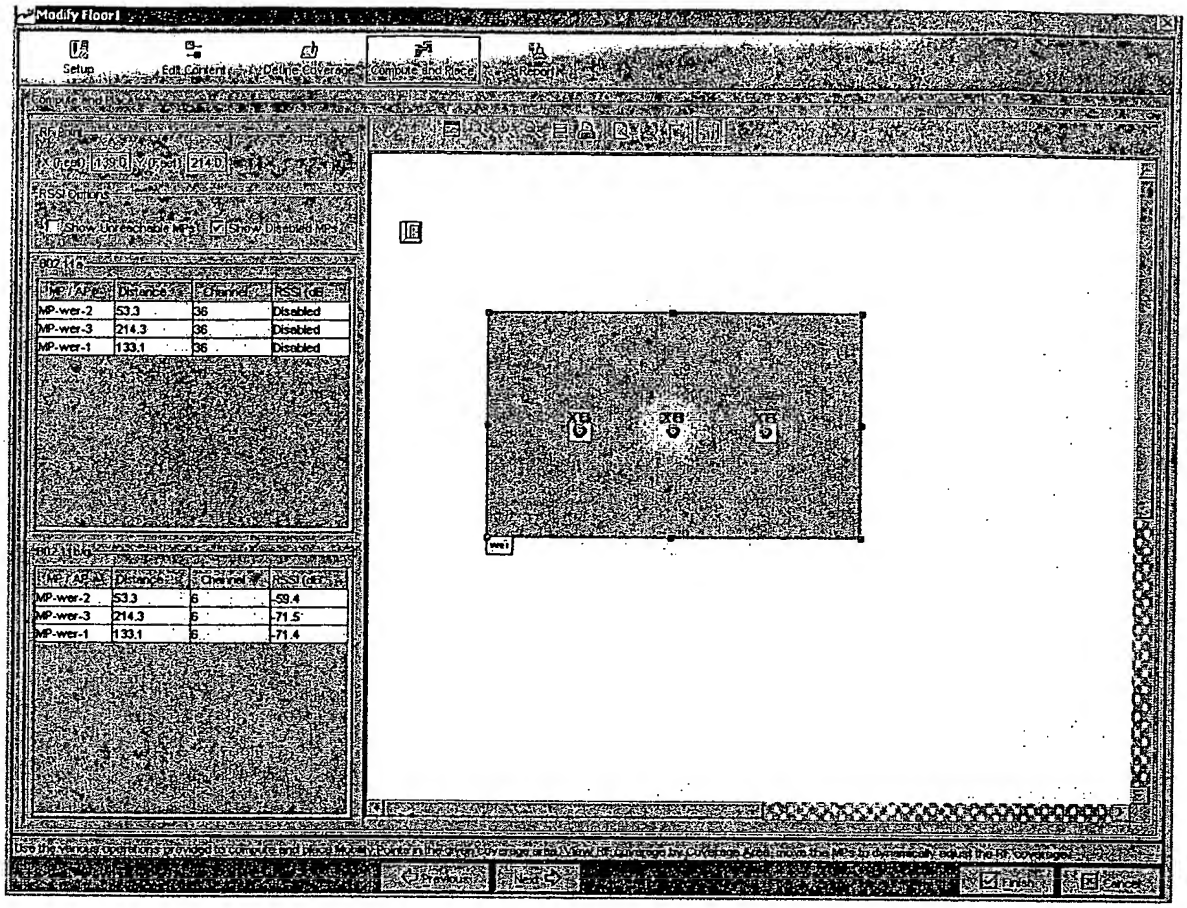
To draw contours to depict RF coverage for a 11g radio, a user must specify if the contour is needed to be shown as 802.11b or 802.11g.

There will be an additional option in the pop-up menu on MP to view 11g RF coverage. If a coverage area is selected, it will draw RF coverage for the technology of the coverage area.

When an 11g coverage area is selected, the user may choose to draw RF coverage for its associated radio in 11b or 11g.

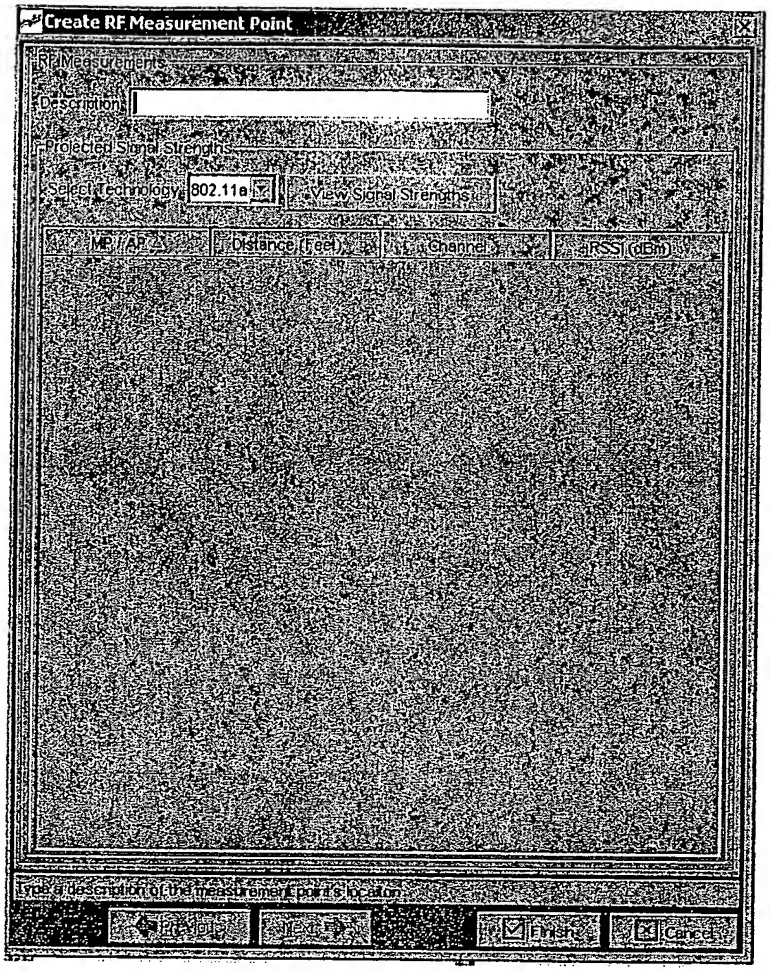
2.1.7 RF MEASUREMENT MONITORING MODE

RF Measyements are analyzed together across 11b and 11g. Following UI will show RSSI readings from 11g radios in the 11b panel.



2.1.8 RF MEASUREMENT POINT MODIFY WIZARD

Similar to reasoning mentioned in the above section, in the modify wizard of a RF Measurement point, the technology option of 11b will be changed to 11b/g.



2.1.9 WORK ORDER CHANGES

Wherever the coverage Area (802.11b) is shown, it will now show "Coverage Area (802.11b/g)". As an example of the work order snippet table for an MP location:

2.1.9.1 .. LOCATION OF MP-QE-3

Model	MX Port (Name:Port)	MX Port (Name:Port)	Coverage Area (802.11a)	Coverage Area (802.11b/g)
MP-252	MX284:P03		Area_a	Area_g

Also, the RSSI readings of 11b and 11g radios will be shown in one table in all the places in the work order.

2.1.10 FLOOR VIEW

A new icon will be added to view the RF coverage of 802.11g areas or radios. An option in the pop-up will be added to view the 11g RF coverage

In the read-only view, the 11g-icon will draw RF coverage for a 11g radio in "pureG"mode. And 11b-icon can be used on a 11g radio to view the RF coverage in "mixedBG" mode.

2.1.11 VERIFICATION RULES

1. Rule to verify that all 11g radios associated with one coverage area are in the same mode of "pureG" or "mixedBG"

2. Rule to verify that the 11g radios associated with a coverage area belong to 1.1 running MX.

3. Rule to verify that the selected Radio profile for a coverage area is the same for the associated radios

4. Rule to verify if the selected MP type is supported in the version of MX that is being deployed to.

5. Rule to verify the following on a coverage area:

    a. If Coverage Area is for 11g and has been forced to use 11g mode, the associated radio profile needs to specify a mode to match the same.

    b. If Coverage Area is for 11g, the associated radio profile needs to specify a mode that is NOT "11b only"

    c. If Coverage Area is for 11b, the associated radio profile needs to specify a mode that is NOT "11g only"

2.1.12 RF DETECTION AND DISPLAY OF ROGUES/KNOWN DEVICE

With the introduction of 11g, RF detection module needs updates to do the following

- allow user to exclude 11g radios
- view 11g discovered devices
- view 11g known devices
- Locate a 11b transmitter, where one 11g is a potential listener. A 11g can listen to 11b only when it is in "mixedBG" mode

2.1.13 CLIENT LOCATION

With the introduction of 11g, client location module needs updates to handle an 11b client being seen by a 11g radio.

2.1.14 NETWORK TOPOLOGY VERIFICATION

With the introduction of 11g, verification of network topology, possibly, needs updates to the new model types and new radio type.

2.1.15 CLI MAPPING/DTD CHANGES

There will be a need to correct the CLI mappings for some commands that will have additional attributes or values.

Action Item: (Product Management) to provide CLI commands changes to incorporate 11g.

Action Item: (Engineering) to decide on DTD changes to incorporate 11g

2.1.16 STATISTICS

There will be additional fields in radio statistics with introduction of 11g. Because of this, the radio statistics display will need updates

Action Item: (MP team) to provide additional fields in radio statistics

2.1.17 IMPACT OF MX VERSIONING

Certain CLI commands or attribute value pairs will not be available for certain versions of MX software. This will impact various planning operations.

1. The user will be able to change the MP type irrespective of the version of MX it is connected to. A verification rule will catch any unsupported mp type errors.

2. Planning tool will create new MX for 11g, if there are no 1.1 MXs in the wiring closet with free ports for MPs.

3. Any 1.0 MX that is uploaded in a network plan for a country-code that is not allowed in 1.0 will be marked as ready to be upgraded to 1.1. This can happen as country code is an optional configuration in basic setting of the box. **Note:** RingMaster must let the user know if there is no 1.1 image present for upgrade to such box, upon next deploy. An example of the work flow will be:

    a. user creates a network plan for the new country code

    b. User uploads an MX that is running 1.0 image

    c. Ringmaster wll accept the configuration from that box and

        i. Change the country code to that of the plan

        ii. Mark the mx for "image and config" upgrade during the next deployment

    d. Upon next deployment, the user will be prompted if there is no 1.1 image present in the image repository

2.1.18 NON-GOAL

11g introduces a mechanism in which a radio can go into "protection" mode to further reduce the throughput. This normally happens when in 11g-environment, a 11b devices are nearby.

Although, MP radio can provide such information in its status, if the radio has gone into protection mode or not, there is currently no requirement in RingMaster to display this information.

3 CONCAVE SHAPE SUPPORT

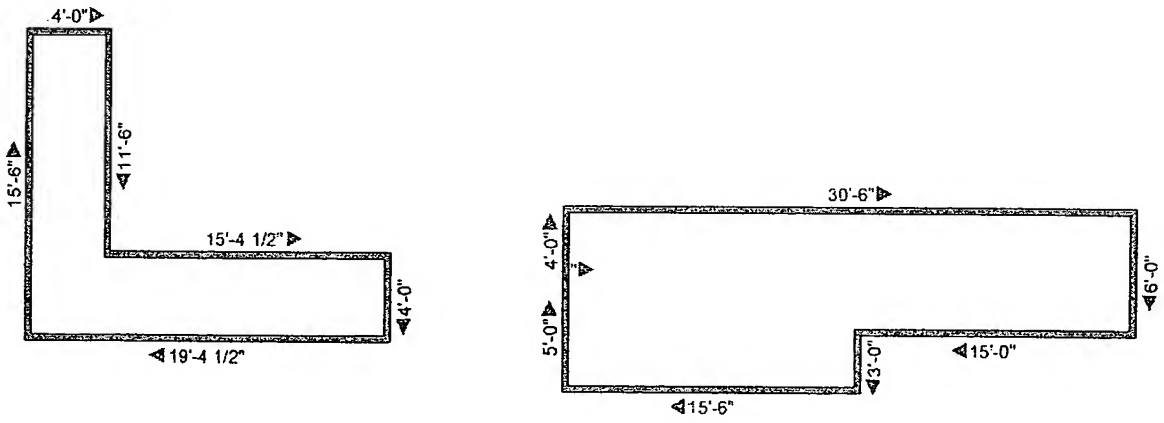
In current implementation of RingMaster, planning tool was unable to handle concave shapes. Also, the shared areas were said to be exactly overlapping each other. Here, we try to solve both of these issues to make the planning tool less restrictive.

The support of this feature will be able to handle the following

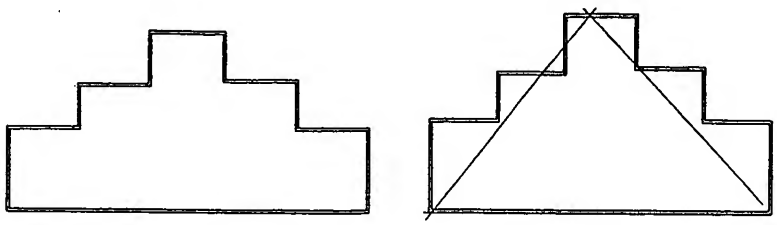
- Concave shaped coverage areas
- Shared coverage areas to use shared MPs only in the overlapped areas

A concave shape is one where any internal angle of the shape is greater than 180 degrees. The user will be able to draw this kind of shape and the planning tool will be able to handle coverage based computation and placement of the APs.

Some examples of concave shares are as follows:



Caution will still have to be taken as to how many such concave angles are provided in the coverage area that is drawn. As an example, if the floor plan does look like the one shown below, a triangular coverage area might end up giving a better result. Geometrically, planning tool will be able to handle any shape, however, more complex concave shapes might end up in slow performance and high number of AP count.



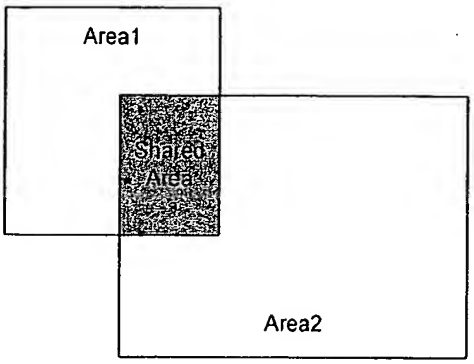
This restriction might not exist by the time this functionality is implemented. But, it is brought out here as possible caution point for planners.

3.1.1 DECOMPOSITION OF A CONCAVE SHAPE

An appropriate algorithm will be chosen to solve this issue.

3.1.2 SHARED COVERAGE AREAS

The user will be able to share MPs across coverage areas that are not completely overlapping each other. As an example, the user will be able to draw the following two coverage areas and then mark it shared. The planning tool will share the MPs only in the overlapped area and compute and place MPs in the unshred area based on the constraints.

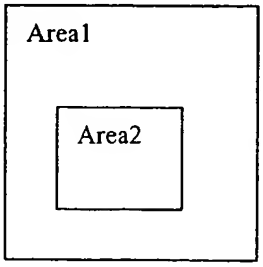


The scope of shared areas still stands as follows:

- No two Coverage areas of the same technology can be shared
- Coverage area for 11b and 11g cannot share MPs
- When placing APs in shared areas, all APs are assumed to be dual-radios
- Any dual-radio mp that belongs to each coverage area and is not locked is a potential candidate to be placed in the shared area.

In addition to the above constraints, following additional rules will apply:

- If the shared area is 90% or more overlapped, then the planning tool will assume the entire area to be overlapped and use the coverage area for 11a as the basis.
- If one coverage area is completely inside the other coverage area, they will not be considered as shared areas. An example is shown in the following picture.



4 MOBILITY ACL SUPPORT

Needs to be defined what this requires.

5 WPA SUPPORT

Wi-Fi Protected Access is a specification of standards-based, interoperable security enhancements that strongly increase the level of data protection and access control for existing and future wireless LAN systems.

Currently, the securirty mode is assumed to be Legacy WEP and the authentication mode is assumed to be 802.1X. In addition, the user may define WEP keys 1...4 at the Radio Profile. Such configuration is apploed to all radios associated with that radio profile.

Enhancing on the same lines, the following additional choices will be available on a Radio Profile.

5.1.1 INFORMATION MODEL

Following security modes will be supported by future releases of MX and MP. Each security mode has certain constraints on the following types of information

1. Authentication mode (Multi-select)
    a. 802.1X
    b. PSK (Pre-Shared Key)
2. Encryption mode: (Multi-select)
    a. TKIP
    b. AES
3. Keys
    a. 4 40-bit or 128-bit Keys for WEP
    b. 1 63-char Key for PSK.

Note: PSK may be defined per MAC-user as well. However, it is not yet decided what route of PSK will be allowed in Trapeze mobility system.

4. Counter Measure Time

    a. the counter measures are spawned when the Message Integrity Check ("MIC") is triggered under conditions defined by the WPA specification. Default: 60 seconds.

5.1.1.1 LEGACY WEP

This will be supported for backward compatibility. It is similar to what exists in release 1.0. The user may choose to specify 40bit or 128bit WEP keys. A total of 4 WEP keys may be defined.

The authentication mode is always 802.1X. Hence, this information is not available for the user to modify

5.1.1.2 WPA ONLY

This type of security mode involves the following:

- Allows any combinations of Authentication modes
- Allows any combination of Encryption modes
- If the Authentication mode is PSK, the user may specify a key using the allowed valid characters
- No WEP keys need be defined.

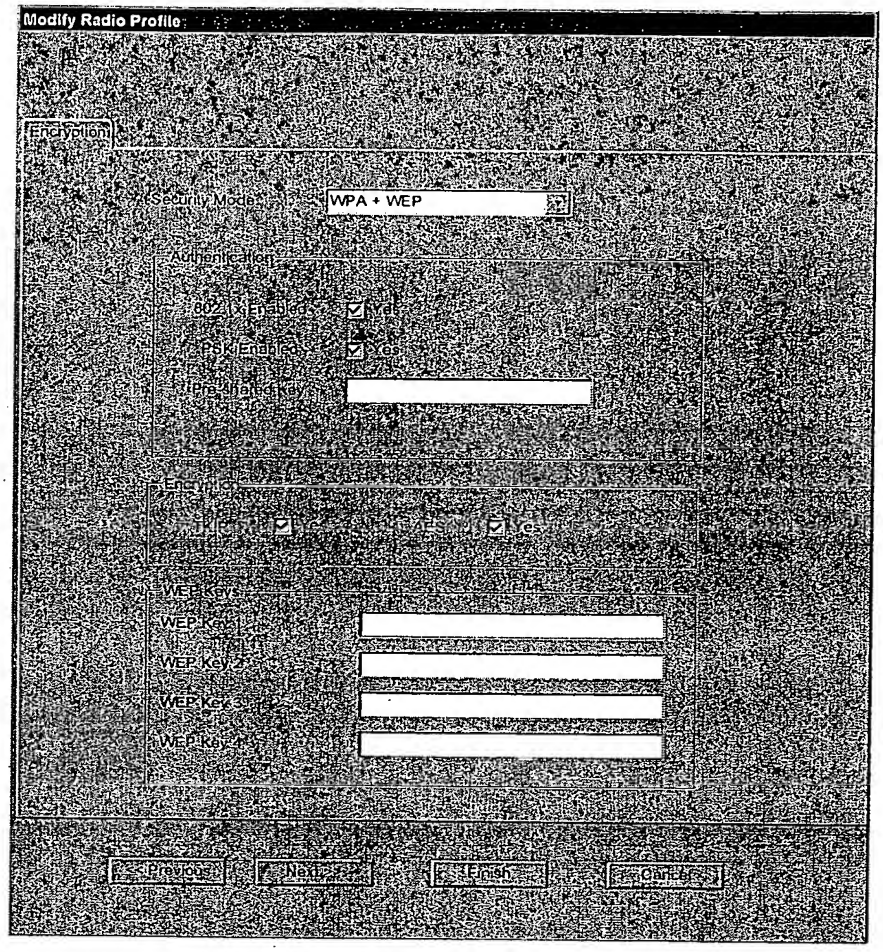
5.1.1.3 LEGACY WEP + WPA

This type of security mode allows clients that talk Legacy WEP or WPA. This involves the following:

- Allows any combination of Authentication modes. This is applicable to WPA only
- Allows any combination of Encryption modes. This ia applicable to WPA only
- A PSK key may be defined if PSK is selected.
- Upto 4 WEP keys may be defined for use of Legacy WEP.

5.1.2 USER INTERFACE

The UI screen will look something like this on the Encryption page when editing a Radio Profile



Summary of Modes:

Mode	Authentication	Encryption	Keys
WEP	802.1x checked and disabled PSK disabled	TKIP and AES disabled	WEP Key1..4 enabled Pre-shared Key disabled
WPA	Both enabled, by default, only 802.1X checked	TKIP and AES enabled, none checked by default	WEP Key1..4 disabled Pre-shared Key enabled, only if PSK is checked
WPA + WEP	Both enabled, by default, only 802.1X checked	TKIP and AES enabled, none checked by default	All keys enabled, PSK is enabled only if PSK is checked

5.1.3 FAULT / EVENTS LOGGING

Useful information that may be obtained from the radio w.r.t. the security is when the radio has gone into performing counter measures. This normally happens when the radio has been hacked into. What the radio does is disassociate all clients and not associate any client for a period of time that can be configured.

It is suggested that a trap be defined in the MX that may be received by NMS monitoring applications, like HP Openview.

Action Item: (MX Team) to confirm if there is an addition of a FACILITY with introduction of WPA.

If the above action item results in an additional Facility, RingMaster will have minor changes to the preference panel.

5.1.4 STATISTICS

New statistics will be defined when WPA is implemented. Currently, no new statistics are defined. If new statistics are implemented, there will be certain changes to Radio based statistics.

Action Item: (MP team) to define new statistics for WPA, if applicable

5.1.5 VERIFICATION RULES

1. A Rule to verify that if an Authentication mode of PSK is selected, a non-empty PSK key is specified.

5.1.6 CLI MAPPING / DTD CHANGES

New commands will be added and this will require mapping to show changes in RingMaster

Action Item: (Product Management) to provide new CLI commands

Action Item: (Engineering) to decide on DTD Syntax to exchange this configuration

5.1.7 IMPACT OF MX VERSIONING

Certain CLI commands or attribute value pairs will not be available for certain versions of MX software.

6 SOLARIS OS SUPPORT

The Solaris Operating System requires the installer to be updated to handle installation as well as other environment updates.

The default location for RingMaster on Solaris will be: /opt/trpz/ringmaster

The sub-directory structure under the install directory will be the same as Windows.

All user and system preferences are stored **[to be figured out where they go but it will be significant]**

[more flushing out required]

7 HP OPENVIEW INTEGRATION

7.1 OVERVIEW

Here is a brief list of features that will be implemented in Release 1.1:

- Installation of integration files
- Menu and Toolbar Integration
- Symbol Integration

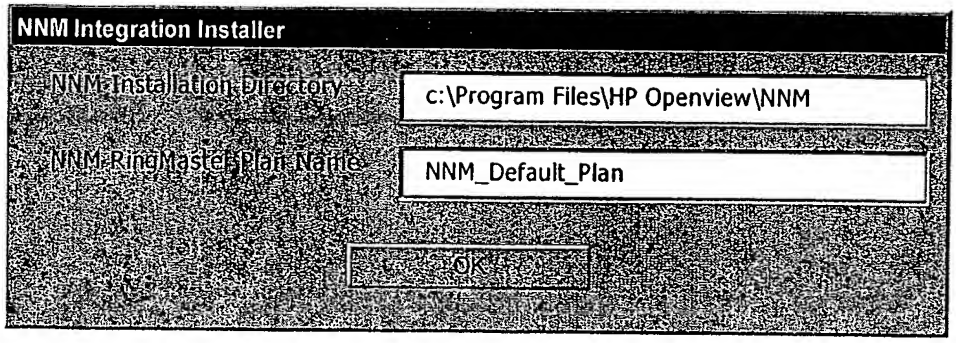
7.2 INSTALLATION OF INTEGRATION FILES

There will be a separate installer to install NNM integration files. At the end of RingMaster installation user will be prompted whether he would like to install NNM integration module. User can proceed with NNM integration installation or come back at a later time and run this installation. RingMaster has no pre-requisite of NNM to be installed before it can be installed.

Pre-requisite to run NNM integration installer

- Need to have installed NNM 6.4 or later version
- Need to have admin privilege to run the installer
- OS supported – Windows XP, Windows 2000, Solaris 8 and 9 on SPARC (No x86 support running Solaris)

During installation installer will try to get the path for NNM from OV_MAIN_PATH environment variable and will prompt to the user to confirm or provide the right location. User will also be asked to enter default plan name that need to be used when RingMaster is launched from NNM. This plan name is inserted as an argument in the places where RingMaster is invoked in the application registration file. User needs to edit the registration file if he wants use a different plan.



Following files will be installed

- Application Registration File
  - UNIX - /etc/opt/OV/share/registration/$LANG
  - Windows – install_dir\registration\%LANG%

- Symbol Registration File
    - UNIX - /etc/opt/OV/share/symbols/$LANG
    - Windows – install_dir\symbols\%LANG%
- Bitmap for the switch
    - UNIX - /etc/opt/OV/share/bitmaps/$LANG
    - Windows – install_dir\bitmaps\%LANG%
- MIB Files (Not sure of the exact location on UNIX)
    - UNIX - /etc/opt/OV/snmp_mibs
    - Windows – install_dir\snmp_mibs

Following files need to be modified during installation

- HPoid_to_sym - This file provide applications with a mapping from sysObjectID to default symbol class and type.
    - UNIX - /etc/opt/OV/conf/oid_to_sym
    - Windows – install_dir\conf\oid_to_sym
- ovw_fields – This file contains vendor specific information
    - UNIX - $OV-FIELDS/c
    - Windows – install_dir\fields\c
- snmp_fields – This file contains SNMP agent information
    - UNIX - $OV-FIELDS/c
    - Windows – install_dir\fields\c
- Oid_to_type – This file provides NNM with mapping from sysObjectID to default object type
    - UNIX – etc/optOV/conf
    - Windows – install_dir\conf

Once above files are updated following commands need to run to make changes effective. Following commands need to be run at the end of installation only if NNM is running. Before running following command we need to convey to the user, for the integration to take place NNM need to be re-launched and get a confirmation whether he wants installation to restart NNM.

Issue: How do we find out whether NNM is running?

ovw -fields

exit ovw

ovstop netmon

ovtopofix –u –o <sysObjectID>

ovstart netmon

ovw

- If installation fails for any reason all the changes done during installation should be reverted back to original state.
- At the end of installation PATH environment variable need to update with RingMaster executable's directory so that NNM will not have any problem launching RingMaster.
- After successful installation if user runs installation again old files should be overwritten and duplicate entries should not be created in the files that are modified during installation.
- During RingMaster installation there is no pre-requisite for NNM to be installed.

7.3 UNINSTALL

- There will be separate uninstall program that user can run anytime to uninstall NNM integration components.
- During uninstall of RingMaster user will be prompted whether he wants to uninstall NNM integration files. Upon user confirmation NNM integration uninstall will be launched to remove all the files that were copied for NNM integration.

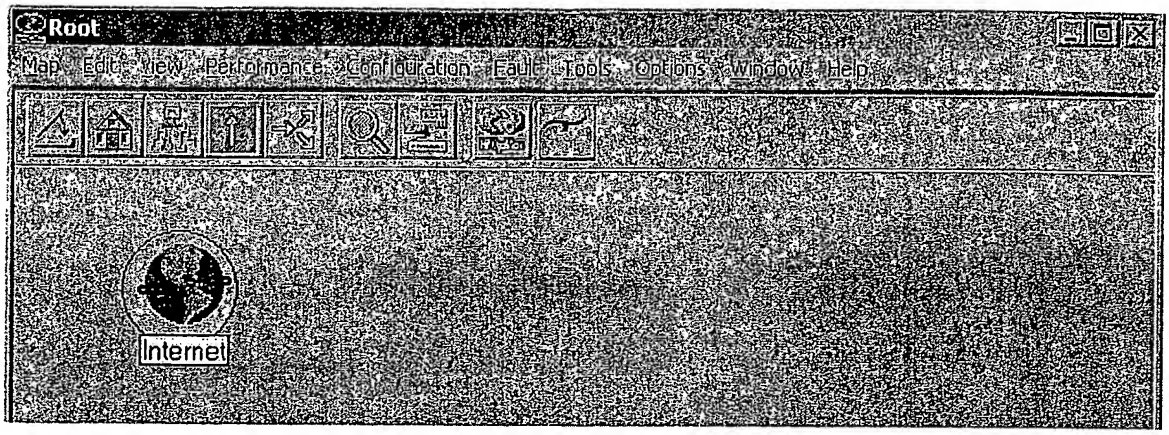
7.4 MENU AND TOOLBAR INTEGRATION

Menu and Toolbar integration is done using Application registration file. This registration file is copied to proper place during installation of NNM support. This file is loaded by NNM and parsed when NNM is started. When NNM begins initialization, it searches in various pre-defined directories for registration files. For every application of symbol type registration file found, NNM opens and parses for correctness. If the entry is valid then NNM performs appropriate operation (for example, adding a menu item in NNM menu structure or adding a button to the toolbar...etc).

7.4.1 TOOLBAR INTEGRATION

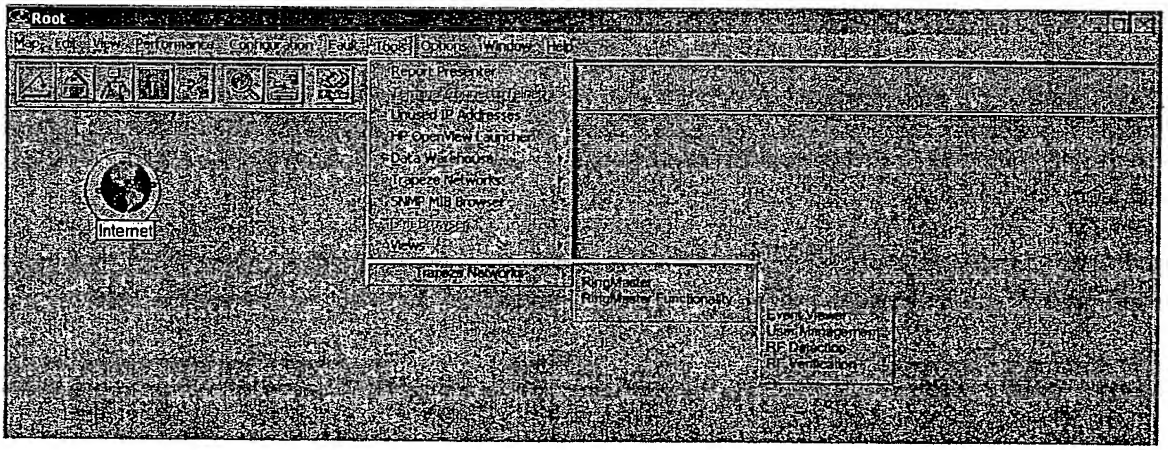
There will be a toolbar button to easily launch RingMaster for NNM. Clicking on this button launches a new instance of RingMaster if there is no instance running. To support this integration on UNIX a pixmap of 24 pixels need to be stored in $OV-BITMAPS/$LANG/toolbar and for windows a bitmap of 16 pixels need to stored in install_dir\bitmaps\$LANG\toolbar directory.

When RingMaster is launched from NNM we can open a default plan. This plan name can be an environment variable or stored as a part of system preference. If a default plan is not provided then user is given an option to create a new plan or open an existing plan.



7.4.2 MENU INTEGRATION

There will be a menu item under Tools menu to launch RingMaster or specific sub functionality (Event Viewer, RF Detection...etc) of RingMaster from NNM. When sub functionality is selected then RingMaster is launched first then selected sub functionality window is shown. Selecting sub functionality menu item from NNM has no effect if there is an instance of RingMaster running. There can be only one instance of RingMaster running at any time.



7.5 SYMBOL INTEGRATION

Symbols are graphical representation of objects in NNM. Symbols can either be icon or connection symbols. Icon symbols represent network or system management elements while connection symbols represent connection between elements.

We will be supporting only icon symbols. These symbols will represent MX. Symbol integration is done using symbol integration files. Each symbol is identified by its symbol type. Symbol type is defined by a symbol class/subclass pair. Symbol class defines the symbol category while subclass defines a particular element within that class.

7.6 COMMAND LINE SUPPORT IN RINGMASTER

RingMaster need to support command line arguments that are passed when it is launched from NNM. One of the arguments that are passed to RingMaster from NNM is default plan name that needs to be

shown when RingMaster comes up. Other argument could be sub functionality that needs to be shown after opening the plan.

We can use following command line flags to identify the arguments passed

- -plan <planName> : -plan flag indicates that following argument is the name of the plan that needs to be opened [if name is empty or null don't open the plan]
- -function <sub functionality> : -function flag indicates that following argument is the sub functionality that needs to be launched after opening the plan.
- Valid sub functionality values are :
    - 1 – Event Viewer
    - 2 – User Management
    - 3 – RF Detection
    - 4 – RF Verification

8 TRANSACTION MANAGEMENT

The application contains multiple background managers that require access to model data. But, these background managers cannot safely use the TxnController as an wizard/action may be in progress. With the current TxnController design, even the act of parsing XML from an external source while another operation is in progress, can corrupt the state of the model. And, for time critical data like statistics and status waiting for the wizard/action to complete is not an option.

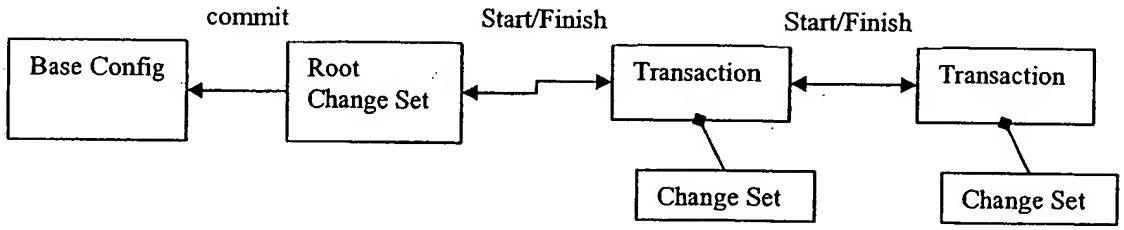
To work around this design deficiency background managers have attempted to use their own TxnController. This scheme works for parsing simple data, but falls apart when references and other relations need to be built as the parsing fails if the relation cannot be consummated. A temporary patch to solve this was to cache any needed objects in the background managers TxnController. This caching is quite expensive as a large part of the model is being replicated in the background manager (consider status collection which builds queries for all ports, APs, radios, etc.)

The proposed solution is to augment the TxnController to use a database technique called MROW (multiple readers, one writer). MROW is desirable as it allows for concurrency without complete serialization. The following sections provide an overview of how MROW can be fitted into the current TxnController with minimal impact to other clients.

8.1 USER VISIBLE FEATURES AND CHANGES

<This section needs to describe all of the user visible features that have to be modified to use the new infrastructure and ultimately what QA need to re-test>

8.2 CURRENT TRANSACTION FLOW

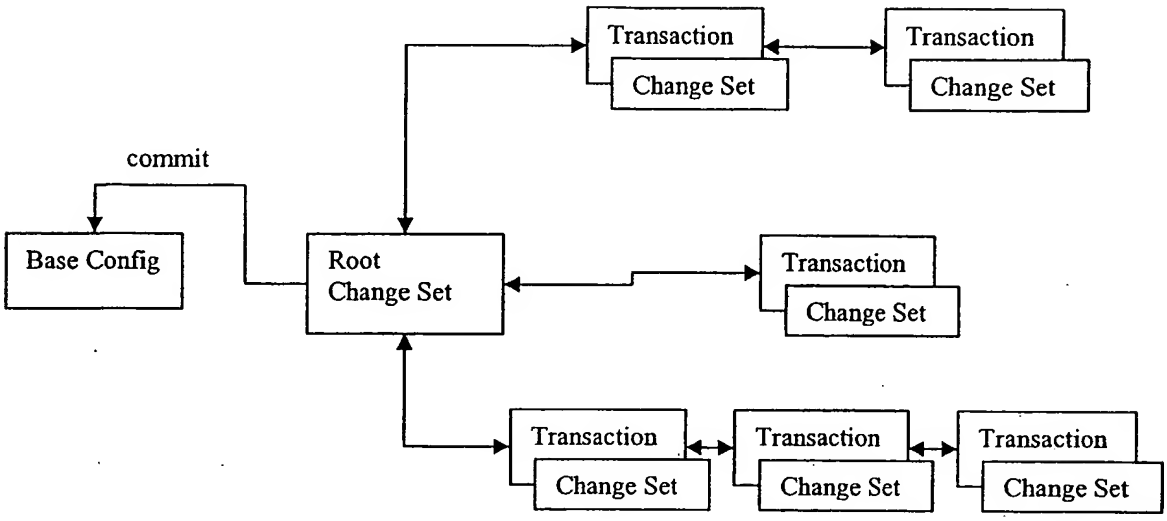


The figure above depicts the flow of a transaction with the current TxnController. As soon as any client starts a transaction, that transactions change set is visible to any other client wants to use the txn controller. This implies that no other client can use the transaction controller in isolation from the state of the original client. This is not good as it violates the "I" (isolation) in ACID, and also forces serialization of read & write operation across clients.

However, in the case of nested wizards isolation of state is not needed as each wizard wants to build on the model state of the prior wizard. This is also true, when one part of the application wants to pass-in its

state to some other part (like a common method.) Hence, any proposed change must allow for sharing of state, as well as isolation of state, depending on the needs of the application.

8.3 PROPOSED FLOW



Conceptually, the proposed change is to allow a multiple transaction chains to be branched on top of the model (base configuration & change set.) The state of each transaction chain is isolated from the other transaction chains.

A client can pass its current state to some other part of the application, allowing nesting of operations that build on prior state.

When a chain completes and its changes are to be merged to the Change Set (CS), or a commit operation is to be performed, a lock must be taken. Otherwise there is no locking or synchronization overhead.

This will allow background managers to start their own transaction chains and safely work in isolation from the rest of the application. Once they perform the necessary tasks the background managers can simply cancel their transaction chain, as they do not need to write to the model.

8.4 PROPOSED DESIGN DETAILS

The concept of isolated transaction chains solves the problem. But these need to be implemented without disrupting the current clients of the TxnController. How this can be achieved is discussed below:

8.4.1 TXN CONTROLLER SPLIT

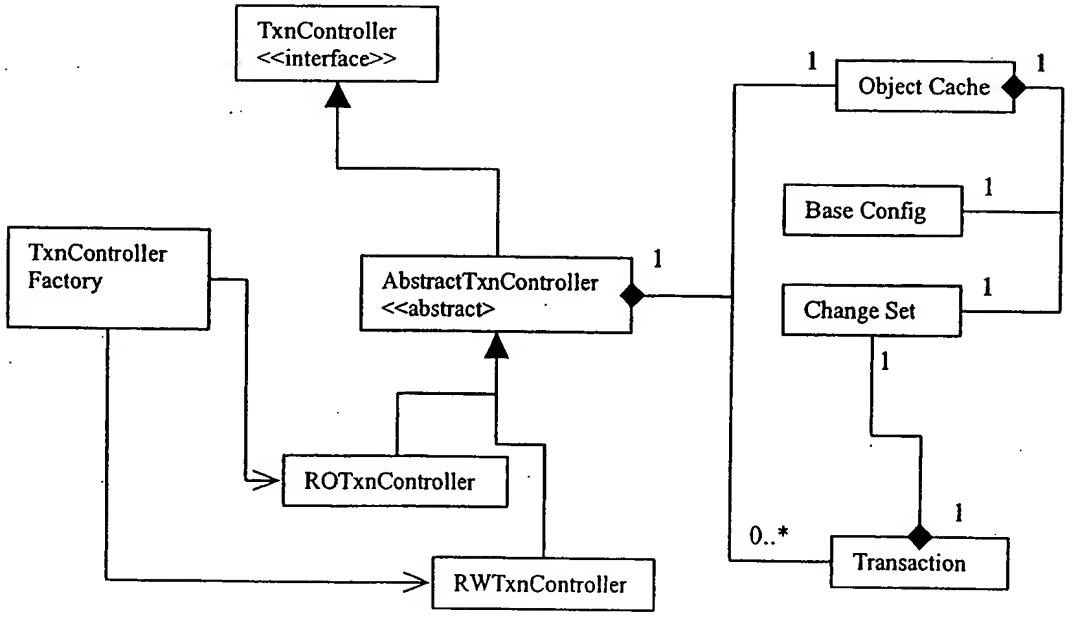
Currently the TxnController interface is implemented by a single class, the TxnControllerImpl. The TxnControllerImpl is instantiated and stored in the NmsFrame, and all other application modules access the TxnControllerImpl via the frame, and use the interface TxnController. As mentioned before, background managers contain a separate instance of the TxnControllerImpl.

The proposed design is to partition the TxnControllerImpl into two separate roles:

- a DataStore that contains the current model base and pending changes
- TxnController that can be used to manage multiple transaction chains

With the proposed design, the impact to the existing application is minimal. Using a factory, the NmsFrame will obtain an instance of the RW_TxnController. All application modules that use the "getTxnController" method will access this instance, and operate on it as before.

Background managers will use a factory to get an instance of a RO_TxnController. They can use this to read application state and safely parse network updates while wizards are active.



8.4.1.1 OBJECT CACHE

The ObjectCache is a subset of the current TxnControllerImpl. It takes over the role of keeping the common model state i.e. the base config and the change set, and is responsible for synchronizing changes to the model state.

The ObjectCache is ignorant of the currently open transactions. The ObjectCache allows multiple transaction chains to be active on top of the shared view of the data.

The ObjectCache is not visible outside of the txn controller package. It is only accessible via the TxnController.

8.4.1.2 READ-WRITE TXN CONTROLLER

The RWTxnController is a TxnController that allows changed state to be written into the RootTxnController. The RWTxnController fetches objects from the RootTxnController as needed. As changes are being made, the object is then cached within the RWTxnController itself.

When the last transaction is finished, the RWTxnController invokes a merge into the Data Store. While this is in progress, no other operations can be performed on the DataStore i.e. it is locked.

The RWTxnController also supports the "commit" call which allows model data to be moved from the Change Set to the Base Config.

8.4.1.3 READ-ONLY TXN CONTROLLER

The ROTxnController allows object level modifications, but does not allow any of these changes to be merged back to the Data Store. When a final finish is done, or a commit is invoked, any changes made in the read-only txn controller are discarded.

This implies that the ROTxnController is useful for making temporary changes. For example, when parsing device stats/status a ROTxnController can be used and once the stats objects are created client obtains and caches them as needed.

8.4.1.4 TXN CONTROLLER FACTORY

When the application is launched a Object Cache must be created, and a Txn Controller Factory must be seeded with the Data Store. The NmsFrame will cache an RWTxnController which will be used for all model changes.

The factory will be used to create multiple ROTxnController objects. Initially we can restrict the factory to produce a single RWTxnController as this helps avoid adding complex logic to handle optimistic or pessimistic object-based locking (see "What if we needed multiple writers...")

8.4.2 COORDINATING WRITES

MROW does not prohibit multiple writers. It requires writes to be coordinated across clients so that only one client is allowed to write at a given instance in time. This is typically done using locking. There are two common variants:

- Optimistic locking: where there is an initial presumption that conflicts will *not* occur, and so no locking takes place until changes are completed and ready to be merged to the data store.

- Pessimistic locking: where locks are granted up front at various levels of granularity and while a lock is held, all other clients pend on it. This scheme is typically based on a timed lock.

There are pros-and-cons to either approach. Optimistic locking is easier on the clients, but requires more complex merge logic. Depending on the implementation details and the order in which clients finish their transactions, it can introduce some timing inconsistencies in the data (unless it rejects a merge based on a conflict.) Pessimistic locking requires more synchronization and needs clients to deal with locks and more importantly being denied locks. But, it eliminates any potential for inconsistencies.

In the current application all modules that need to update the model will coordinate their operations via the NmsFrame (using get/set busy methods.) Hence there is really no need for multiple writers. To leverage this, instead of requiring any synchronization or merge logic, we can enforce the single writer by having the factory/frame only contain a single instance of the RW_TxnController. If & when needed, this scheme can be seamlessly extended to support multiple writers and the proper write coordination logic.

8.5 FUTURE APPLICATIONS....

8.5.1 WHAT IF WE WENT CLIENT/SERVER

The proposed design can adapt well to a distributed model. Each client can have one or more of its own read-only or read-write TxnController instances and the Data Store can reside in the server.

By performing all object operations locally, and without any synchronization overhead, clients can be extremely efficient. When changes are ready to be merged, an entire Change Set can be transferred back to the server.

8.6 DELIVERABLES & ESIMATES

8.6.1 GENERAL

With the new design, all background tasks can be safely performed using a ROTxnController. A client that wants to process a network response in the background can use a ROTxnController to parse the XML and create RTime objects. Once these are parsed they can be cached in the StateMonitor or propagated back to other modules, like in the case of stats collection.

The DeviceStatManager, OperStatusPropogator and Client management module need to be updated to do this.

8.6.2 CLIENT MANAGEMENT

The client management module uses a mix of dynamic and configuration data. For background tasks it needs to be updated to use a ROTxnController (like the DeviceStatManager, etc.)

Here is an initial analysis of the changes necessary for this module:

- We need to have a ClientMgr singleton object which maintains a ROTxnController. ClientMgr will be instantiated whenever a user opens a new plan and disposed whenever a plan is closed.

- ClientMgr will open a long transaction using a ROTxnController instance, and will listen to APRadio delete events since it actually establishes the reference relations from current user location to the AP radio. If a radio is deleted, the ClientMgr's ROTxnController will need to be updated.

- ClientMgtPanel and FindUserWizard will both use this ROTxnController to do create and delete or modify of the user sessions and user locations whenever we perform find Users, or polling users from background

- ClientMgtPanel will no longer need to cache the data, and it will use the ROTxnController to update the user session and user location data. And when it is doing background polling, it will not need to set the frame to be busy since it is operating on a different transaction controller.

- Since ShowUserLocation() method in ClientMgtPanel sends FloorLayoutEventData to FloorMdlView, this event data will need to have slight interface change to pass in session label, rssi, and AP radio key (which MX, MP, and Radio Slot) instead of passing session id and radio id. This is because the session now is no longer created in the main RWTxnController; we need to de-couple the usage of the id.

8.6.3 ESTIMATES

(Estimates include unit testing)

1. Infrastructure changes (with single RWTxnController support) – 5 days
2. Devif - DeviceStatManager changes – 2 days
3. Oper Status Propagator changes – 1 day
4. Client management/Find Client - (3 days)

9 VERSIONING

Moving forward Ringmaster will need to support multiple versions of software and maintain a level of compatibility between them.

9.1 XML CONVERSION

9.1.1 DEVICE DTD COMPATIBILITY

9.1.1.1 PATCH RELEASES

For patch releases a DTD needs to be backward compatible. That implies that a 1.0.x+1 DTD must be able to validate a 1.0.x XML.

In order to achieve this some rules must be followed:

- Only optional attributes can be added
- Only optional elements can be added, and they must be at the end i.e. no change in document order for existing elements.
- No other changes are allowed

9.1.1.2 MAJOR/MINOR RELEASES

Need to clarify what is supported. Possible changes???

- attribute is added
- attribute is removed
- attribute is modified
    - Type changes
    - Range changes:
        - Enum list extended
        - Enum list shortened
        - Numerical ranges?
        - String lengths?
- element is moved
- element is removed

- element is added

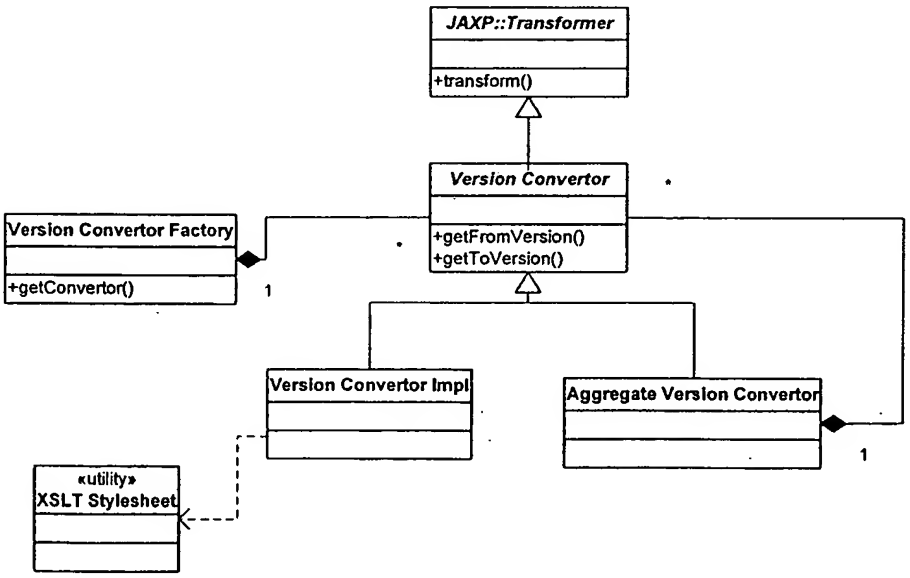
9.1.2 VERSION CONVERTORS

The proposal is to develop a set of version converters which will be implemented as JAXP Transformer instances (see JAXP documentation) to convert between various versions of the XML. Each transformer will transform the XML between two immediate versions. For future conversions across multiple versions transformers can be chained together.

A transformer will typically be implemented by an XSLT stylesheet. Each stylesheet can consist of multiple templates (templates are like procedures in XSLT) for various conversions.

A transformer can also have an implementation that converts directly between two DOMs i.e. it is not required to be a XSLT stylesheet.

For example, the policy data may change between 1.0.0 & 1.0.1. To handle this, a "1.0.0 to 1.0.1" transformer will be created and registered in a Transformer Factory. When a client module encounters a 1.0.0 XML and wants to convert it to 1.0.1, it will lookup this transformer and will run it to produce a 1.0.1 XML.



Depending on how complex the conversion is we can also support an aggregation of converters for a single conversion. So for example, assume that a conversion has the above mentioned policy changes and also has a new AAA/userglob hierarchy. We can develop independent Version Converters for each conversion, and then somehow aggregate them together for the full conversion. If we find the conversion getting too complex, this approach may help in breaking it into simpler pieces.

10 RULES SUPPORT

We have added additional support for Rules Check in RingMaster release 1.1. For details, please refer to the Rules-Spec document. Here is only a summary of a list of rules added for release 1.1:

1. Accounting for MAC Network Access is not supported. (Error)
2. AAA User/UserGroup, Mac User/Mac UserGroup Attributes validation:
    a. Moblity-Domain Profile should exist in the device (Warning)
    b. Service-Type needs to be numberic value (1-11) (Error)
    c. Encryption-Type needs to be numeric value (0-64) (Error)
    d. Session-Timeout needs to be non-negative number (>0) (Error)
    e. Idle-timeout needs to be non-negative number (0-65535) (Error)
    f. Filter-id needs to postfix with either *.in or *.out (Error)
3. AAA Radius Server "key" should be set if Radius Default did not set the default for "key" (Warning)
4. AAA Radius Server can not have ip addresss as 0.0.0.0 (Error)
5. Mobility-Profile should contain at least 1 port group reference if the mode is defined as Selected. (Error)
6. ManagementServices Sys Log should have maximum of 4 log servers. (Error)
7. ACL name should start with alphabetical characaters.
8. ACL name should not contain the following terms: **all, default-action, map, help, editbuffer**

11 MISC CONFIG SUPPORT

We have also added the configuration support in R1.1 for the following (was unsupported in R1.0):

- VR-ARP configuration: One can now configure VR-ARP agingTime, and ARP Entries (hw-addr, and ip-addr), deploy, and review network changes etc.
- Trace-Table configuration. One can now configure Trace-Table, setting up different trace area, levels, deploy, and review network changes etc.

12 EVENT VIEWER ENHANCEMENTS

Several new features will be added to to the event viewer in R1.1

- The user must be provided with the ability to enable or disable the auto-refresh functionality.
- The user must be able to specify AND and OR conditions when specifying text search criteria in the event filters.
- A function to find a string within a message must be provided in the detailed view dialog.
-

13 APPENDIX

13.1 NNM APPLICATION REGISTRATION FILE DEFINITION

Application registration files are used to integrate network and systems management applications with the NNM user interface. Many aspects of an application's integration are defined using an application registration file (ARF). Application registration files provide NNM with important information such as:

- How to integrate the application into the NNM menu and Toolbar structure
- How to invoke the application based on the user's run-time selection of menu items

Ex:

```
Application "RingMaster"
{
    /*
     ** APPLICATION DESCRIPTION
     */
    DisplayString "Trapeze Networks Planning Tool"
    Version "RingMaster 1.0"
    Description {
        "Description....."
    }
    Copyright {
        "Copyright information ....."
    }

/*
** COMMAND BLOCK
*/
/*
 ** Valid Process_flags are Initial, Shared and Restart
*/
```

```
Command  -[process_flags]  "command_name" $environment_variable;


/*

 ** MENU BLOCK

*/

MenuBar <100>  "Tools" _T

{

    <10> "Trapeze Networks"  _z  Context (AllContexts) f.menu
"trapeze";

}

Menu "Trapeze Networks"

{

<100> "Ring Master" _R Context(AllContexts) f.action "ringmaster";

<90> "Event Viewer" _E Context(AllContexts) f.action "eventviewer";

<80> "RF Detection" _D Context(AllContexts) f.action "rfdetection";

<70> "Client Management" _C Context(AllContexts) f.action
"clientmanagement";

}


/*

** TOOLBAR  BLOCK

*/

ToolBarButton <50> @"toolbar/ringmaster.bmp, RingMaster"


Context "AllContexts" f.action  "ringmaster"


/*

** SYMBOL POPUP MENU BLOCK

*/
```

```
    PopupItem <100> "RingMaster"
    Context AllContexts
    TargetSymbolType "Net Device":"Trapeze MX-20 switch"
    f.action "ringmaster";
PopupItem <90> "Event Viewer"
      Context AllContexts
        TargetSymbolType "Net Device":"Target MX-20 switch"
         f.action "eventviewer";
PopupItem <80> "RF Detection"
      Context AllContexts
        TargetSymbolType "Net Device":"Target MX-20 switch"
         f.action "rfdetection";
PopupItem <70> "Client Management"
      Context AllContexts
      TargetSymbolType "Net Device":"Target MX-20 switch"
      f.action "clientmanagement"

/*
** ACTION BLOCK
*/
   Action "ringmaster"
   {
   Command -shared  "ringmaster" -p $NNM_RM_DEFAULT_PLAN ;
   }
   Action "eventviewer"
   {
   Command -shared "ringmaster" -p $NNM_RM_DEFAULT_PLAN -f  "event";
```

```
}

Action "rfdetection"

{

Command -shared "ringmaster" -p $NNM_RM_DEFAULT_PLAN -f
"rfdetection"

}

Action "clientmanagement"

{

 Command -shared "ringmaster" -p $NNM_RM_DEFAULT_PLAN -f
"clientmanagement"

}
```

NNM SYMBOL REGISTRATION FILE DEFINITION

SymbolType "class Name" : "subclass Name"

{

Filebase "symbol_class_icon_base_name";

CursorSize n;

DisplayString "localizable String";

}

Ex:

SymbolType "Connector" : "Trapeze MX-20 Switch"

{

Filebase "trpzmx20";

CursorSize 38;

DisplayString "Trapeze Networks MX-20 switch";

}

In the above symbol subclass definition subclass name needs to be unique.

Filebase defines the base name for a symbol subclass. It is provided in a file with the format filebase.size.extension Symbol class icon can be an X bitmap or X pixmap. Pixmap is a supported format for UNIX and Windows.

Bitmap definition is composed of two parts filebase.size.p (the bitmap) and filebase.size.m (the bitmap mask). Pair of bitmap/bitmap mask file pair should be provided for each bitmap size. Recommended symbol subclass icon sizes (in pixels): 20X20, 26X26, 32X32, 38X38, 44X44 and 50X50. All icons for a subclass must be of the same format. Pixmap definition consists of simply filebase.size.pm because the mask is defined in the pixmap (Need to investigate whether GIF or JPEG can be used instead of pixmap or bitmap files)

CursorSize entry defines the size of the bitmap to be used as the cursor. CursorSize is also used during drag and drop operation. Recommended cursor size is 38X38.